EP 17381 (3)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : H04J 3/02 — A1

(11) International Publication Number: WO 95/26595

(43) International Publication Date: 5 October 1995 (05.10.95)

(21) International Application Number: PCT/US94/07864

(22) International Filing Date: 11 July 1994 (11.07.94)

(30) Priority Data:
219,665 29 March 1994 (29.03.94) US

(71) Applicant: SCIENTIFIC-ATLANTA, INC. [US/US]; One Technology Parkway - South, Norcross, GA 30092 (US).

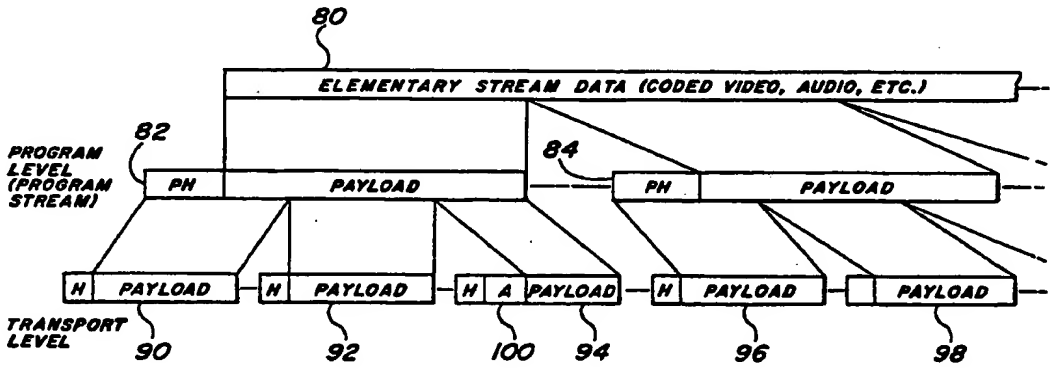
(72) Inventors: WASILEWSKI, Anthony, J.; 10680 Wren Ridge Road, Alpharetta, GA 30202 (US). LOGSTON, Gary, L.; 441 Engle Drive, Tucker, GA 30084 (US).

(74) Agents: ROCCI, Steven, J. et al.; Woodcock Washburn Kurtz Mackiewicz & Norris, One Liberty Place, 46th floor, Philadelphia, PA 19103 (US).

(81) Designated States: AU, CA, JP.

Published
With international search report.

(54) Title: PACKET ALIGNMENT METHOD AND APPARATUS FOR SIMPLIFYING PARSING AT A DECODER IN A PACKET-BASED COMMUNICATIONS SYSTEM



(57) Abstract

A method of packetizing a data stream (80) for transmission comprises the steps of (a) segmenting the data stream (80) and inserting successive segments of the data stream into a plurality of successive program packets (82), each program packet having a header followed by a payload section; (b) for each of the successive program packets, (i) inserting successive segments of the program packet into the payload sections of a consecutive sequence of transport packets (90-98) such that the header of the program packet is aligned with the beginning of the payload section of a first transport packet in the sequence; and (ii) repeating step (b) (i) until the last segment of the program packet is reached. If the last segment of the program packet does not fill the entire payload section of a last transport packet in the sequence, then stuffing bytes are added to the last transport packet to fill out the remainder of the payload section of that transport packet. In this manner, the header of each program packet is always aligned with the beginning of a transport packet payload.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SI	Slovenia
SK	Slovakia
SN	Senegal
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

PACKET ALIGNMENT METHOD AND APPARATUS FOR SIMPLIFYING PARSING AT A DECODER IN A PACKET-BASED COMMUNICATIONS SYSTEM

FIELD OF THE INVENTION

The present invention relates a method and apparatus for packetizing a data stream for transmission to a decoder, and more particularly, to a packet alignment method and apparatus for simplifying the parsing of a transport stream in a packet-based multiplexed communications system.

BACKGROUND OF THE INVENTION

Recently, the International Organization for Standardization (ISO) adopted a standard protocol for combining one or more "elementary streams" of coded video, audio or other data into a single bitstream suitable for transmission. The standard (ISO 13818), hereinafter referred to as the "MPEG-2 Systems" standard, is described in detail in the MPEG-2 Systems Committee Draft (ISO/IEC JTC1/SC29/WG11/N0601, November, 1993) [hereinafter "MPEG-2 Systems Committee Draft"], which is hereby incorporated by reference. An overview of the MPEG-2 Systems standard is provided in Wasilewski, *The MPEG-2 Systems Specification: Blueprint for Network Interoperability* (January 3, 1994), which is also hereby incorporated by reference. The MPEG-2 Systems standard provides a syntax and set of semantic rules for the construction of bitstreams containing a multiplexed combination of one or more "programs." A "program" is composed of one or more related elementary streams. An "elementary stream" is the coded representation of a single video, audio or other data stream that shares the common

timebase of the program of which it is a member. For example, in the context of a subscription television system, a "program" may comprise a network television broadcast consisting of two elementary streams: a video stream and an audio stream.

As discussions concerning the MPEG-2 Systems standard progressed, the MPEG-2 Systems Committee decided that a two-level packet-based multiplexing approach would best serve the objectives of the new standard. In a paper presented by Applicants, on behalf of their Assignee, to the International Standards Organization, entitled "A REVISED Proposal for MPEG-2 Transport and Program Multiplex Syntax", MPEG 93/351 (March 29, 1993), which is hereby incorporated by reference, Applicants proposed a detailed syntax and methodology for such a two-level packet-based multiplex approach. Generally, the two-level multiplex approach proposed by Applicants comprises a first "program" level in which each elementary stream (i.e., the coded representation of one video, audio or other digital data signal) is segmented and inserted into the payload sections of a respective sequence of "program packets" to form a "program stream" for that elementary stream, and a second "transport" level in which the program packets of each program stream are further segmented and inserted into the payload sections of smaller, fixed-length "transport packets". The transport packets containing the program stream data of each elementary stream may then be multiplexed in a time division manner to generate a single outgoing bitstream referred to herein as a "transport stream". Thus, a transport stream comprises a continuous sequence of transport packets, each of which may carry data from one of a plurality of different program streams. Such a multiplexing approach can be used in a wide variety of applications. For example, such a multiplexing approach is particularly well suited for the transmission of audio and video programming in a subscription television system, such as a CATV or Direct Broadcast Satellite (DBS) system.

As Applicants recognized, with any two-level multiplexing approach, a significant amount of parsing will be required in a decoder at a reception site in order to recover the elementary stream data of a program selected by a user at that site. Consequently, methods and apparatuses are needed for generating transport streams that simplify or facilitate parsing of the stream at a decoder. The present invention satisfies this need. Applicants, on behalf of their Assignee, proposed the method of the present invention for inclusion in the MPEG-2 Systems standard, and the method of the present invention, as defined by the appended claims, was ultimately incorporated into the standard.

SUMMARY OF THE INVENTION

The present invention is directed to a packet alignment method for simplifying the parsing of a transport stream in a packet-based multiplexed communications system, and to an apparatus for carrying out the method. According to the method, successive segments of a data stream to be transmitted are inserted into a plurality of successive program packets. Each program packet has a header followed by a payload section. The payload sections of each program packet preferably have variable lengths. A first one of the successive program packets is then received for further packetization. Successive segments of the first program packet are then inserted into the payload sections of a sequence of smaller, fixed length transport packets. The header of the program packet is aligned with the beginning of the payload section of a first transport packet of the respective sequence. Subsequent segments of the program packet are then inserted into subsequent transport packets until the end of the current program packet is reached. If the last segment of the current program packet does not fill the entire payload section of a last transport packet in the sequence, stuffing bytes are inserted in the last transport packet along with the last segment of the current program packet in order to "fill-out" the entire payload section of

that transport packet. A next subsequent program packet is then packetized in a similar manner beginning with a new transport packet. In this manner, the beginning of each program packet is always aligned with (i.e., always occupies) the first byte of a respective transport packet payload.

In a preferred embodiment, every transport packet has a header and a bit is set in the header of every transport packet that carries the header of a particular program packet. As a result, a decoder can parse the headers in a sequence of transport packets to locate the boundaries between successive program packets carried in the payload sections of those transport packets. Preferably, stuffing bytes are inserted in transport packets in the form of an adaptation field.

An apparatus according to the present invention comprises means for segmenting a data stream to be transmitted and for inserting successive segments of the data stream into a plurality of successive program packets, each program packet having a header followed by a payload section; means for receiving one of the successive program packets; means for inserting successive segments of the program packet into the payload sections of a sequence of transport packets such that the header of the program packet is aligned with the beginning of the payload section of a first transport packet of the sequence; and means for inserting stuffing bytes in a last transport packet of the sequence if a last segment of the program packet does not fill the entire payload section of the last transport packet.

Preferably, the apparatus further comprises means for generating an adaptation field for insertion in the payload section of a transport packet. In this case, the means for inserting stuffing bytes is preferably adapted to insert the stuffing bytes in an adaptation field of the last transport packet. In a preferred embodiment, every transport packet has a header and the apparatus further comprises means for setting a bit in the header of the first transport packet

to indicate that the first transport packet carries the header of a program packet.

Additional features and advantages of the present invention will become evident hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing summary, as well as the following detailed description of the preferred embodiment, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings an embodiment that is presently preferred, it being understood, however, that the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

Figure 1 illustrates an application of the present invention to a subscription television system;

Figure 2 is a block diagram of an encoder apparatus for generating a transport stream in accordance with the method of the present invention;

Figure 3 graphically illustrates the packet alignment method of the present invention;

Figure 4 shows additional details of the general content and arrangement of a program packet in accordance with the present invention;

Figure 5 shows additional details of the general content and arrangement of a transport packet in accordance with the present invention;

Figure 6 shows the general content and arrangement of a transport packet containing an adaptation field in accordance with an aspect of the present invention;

Figure 7 is a flow diagram illustrating further details of the operation of the encoder of Figure 2 and a preferred embodiment of the method of the present invention;

Figure 8 illustrates further details of the transport level packetizer of Figure 2 and, in particular, illustrates an apparatus for carrying out the method of Figure 7; and

Figure 9 is a block diagram of a decoder apparatus for parsing an incoming transport stream to extract one or more elementary streams from the transport stream.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to the drawings wherein like numerals indicate like elements throughout, Figure 1 illustrates an application of the present invention to a direct broadcast satellite (DBS) subscription television system 10. It is understood, however, that the present invention is by no means limited thereto. Rather, the present invention may be employed in any packet-based multiplexed communications system wherein it is desirable to simplify the parsing of a transport stream (as defined herein) by a decoder at a reception site in the system.

As shown in Figure 1, the exemplary DBS system 10 comprises an uplink site 12 for generating a transport stream containing a multiplexed combination of audio and video signals and for transmitting the transport stream to one or more subscriber locations, e.g. location 33. The audio and video signals (e.g., Video 1, Audio 1 ... Video N) are input to an encoder 14 via respective input lines 16, 18..24. The encoder 14 operates in accordance with the method of the present invention to generate a continuous transport stream comprising a time-division multiplexed combination of transport packets, each of which may contain data comprising a coded representation of one of the video or audio signals input on lines 16, 18..24. As the transport stream is generated, it is supplied via line 26 to a transmitter 28 and then to a satellite uplink 30 for transmission via satellite 32 to the subscriber location 33. The transport stream is typically transmitted over a particular transponder frequency.

At the subscriber location 33, a receiver 36 is tuned to the particular transponder frequency for receiving the transport stream via a satellite down-link 34. The incoming transport stream is then provided to a decoder 40

via line 38. A subscriber can employ the decoder 40 to retrieve the audio and video signals of a particular program from the incoming transport stream for output to a television set 42 or audio output device 44 at the subscriber location 33.

Figure 2 is a block diagram illustrating further details of the encoder 14 of Figure 1. As shown, the video and audio signals (e.g., "Video 1", "Audio 1" ... "Video N"), which preferably are in digital form, are provided on respective lines 16, 18..24 to respective coders 46, 48..50 that produce coded representations of the respective signals. For example, the video and audio coders may operate in accordance with any well known data compression technique to produce digital compressed representations of the respective video and audio signals. As mentioned in the Background of the Invention, the coded representation of one video, audio or other data signal is referred to herein, as well as in the MPEG-2 Systems Committee Draft, as an "elementary stream".

As further shown in Figure 2, each elementary stream is provided to a program level packetizer 51. For example, the elementary stream for Video 1 is provided to the packetizer 51 via line 16', the elementary stream for Audio 1 is provided to the packetizer 51 via line 18', and so on. The program level packetizer 51 implements the first level of Applicant's two-level multiplexing scheme. As described hereinafter in greater detail, the program level packetizer 51 segments and inserts each elementary stream into a respective sequence of program packets referred to herein as a program stream. Each program packet in a given program stream consists of a program packet header followed by a variable length payload containing the coded data of the elementary stream carried in that program stream. Each elementary stream, in its program stream format, is assigned a unique "Packet ID" (PID). For example, the elementary stream "Video 1" may be assigned a PID of "10", the elementary stream "Audio 1" may be assigned a PID of "23", and so on.

Once generated, the program streams are provided to a transport level packetizer 53 via respective lines 52, 54..56. The transport level packetizer 53 implements the second level of Applicant's two-level multiplexing scheme. In particular, as described hereinafter in greater detail, the transport level packetizer 53 segments the individual program packets of each program stream and inserts successive segments of each program packet into the payload sections of a respective sequence of smaller, fixed-length transport packets. Thus, each program stream is carried in a respective sequence of transport packets. Each transport packet in a given sequence has a header that contains the PID associated with the program stream carried in that sequence of transport packets.

As further shown in Figure 2, the sequences of transport packets formed from each program stream are provided to a transport stream multiplexer 64. For example, the sequence of transport packets carrying the program stream data for elementary stream "Video 1" are provided to the multiplexer 64 via line 58, the sequence of transport packets carrying the program stream data for elementary stream "Audio 1" are provided to the multiplexer 64 via line 60, and so on. The multiplexer 64 operates in accordance with a time-division multiplexing technique to multiplex the individual transport packets of each sequence to form a single outgoing bitstream referred to herein as a transport stream. The transport stream is output on line 26 for transmission (see Figure 1).

A system clock generator 68 generates a common system clock that governs audio and video coding, the program level and transport level packetizers, and the transport stream multiplexer. A control computer 70 controls the overall function of each component of the encoder 14 via control lines 72, 74..76. Additionally, the control computer 70 may generate a number of other transport packets that contain system related information, such as encryption related information used to control access to the elementary

stream data carried in the transport stream. These system related transport packets are provided to the transport stream multiplexer 64 via line 63.

A transport stream therefore comprises a continuous sequence of transport packets, each of which may either carry system related information or data from one of the plurality of program streams.

Figure 3 graphically illustrates the two-level packet-based multiplexing scheme implemented by the encoder 14 of Figure 2, and more particularly, illustrates the packet alignment method of the present invention. As shown, at the program level, an elementary stream 80, which may comprise a stream of coded video or audio data, is segmented and inserted into the payload sections of a plurality of successive program packets, e.g. program packets 82 and 84, to form a program stream. Each successive program packet 82, 84 comprises a program packet header (PH) followed by a variable length payload comprising a successive portion of the coded data of the elementary stream 80. Generation of the program stream is performed by the program level packetizer 51 of Figure 2.

At the transport level, each program packet 82, 84 of the program stream is further packetized into a sequence of transport packets 90, 92..100. Each transport packet 90, 92..100 comprises a header (H) followed by a payload portion that carries the program stream data. According to the present invention, the program packets 82, 84 of the program stream are inserted into the transport packet payloads such that the first byte of the header of each program packet 82, 84 always occupies the first byte of a respective transport packet payload. For example, as illustrated in Figure 3, a first portion of program packet 82, which begins with a program packet header (PH), is inserted in the first transport packet 90 of the sequence of transport packets 90, 92..100. A second successive portion of the program packet 82 is inserted into the next consecutive transport packet 92. As illustrated however, the third or remaining portion of the

program packet 82 is not long enough to occupy the entire payload section of the next consecutive transport packet 94. One way to handle such a situation is to simply fill as much of the payload section of the transport packet 94 as the remaining portion of the program packet 82 will occupy, and then begin filling the remainder of the transport packet 94 payload with the beginning of the next program packet 84. As Applicants recognized however, such a technique is disadvantageous because a decoder that receives this sequence of transport packets 90, 92..100 would have to parse through the payload sections of each incoming transport packet to determine the boundaries between successive program packets. This would unduly increase the cost and complexity of the decoder. The packet alignment method of the present invention provides a simple mechanism for determining the boundaries between successive program packets as they are recovered from their respective sequence of transport packets without the need for complex parsing capabilities.

According to the packet alignment method of the present invention, as illustrated in Figure 3, when the last segment of a program packet, e.g. program packet 82, will not fill the entire payload section of the next consecutive transport packet, e.g. transport packet 94, an appropriate number of stuffing bytes are inserted to "fill out" the payload section. Consequently, the first byte of the header (PH) of the next program packet, e.g. packet 84, will occupy the first byte of the payload section of the next consecutive transport packet, e.g. packet 96. Thus, the method of the present invention ensures that the first byte of each program packet is always aligned with (i.e., always occupies) the first byte of the payload section of a transport packet. As explained hereinafter in greater detail, a program packet header indicator flag is provided in the header (H) of each transport packet 90, 92..98 to inform a decoder whether a given transport packet contains the beginning (i.e., the header) of a particular program packet. A decoder can use the program packet header indicator flags in each transport

packet to determined the boundaries between successive program packets recovered from those transport packets. Thus, the decoder need only parse the transport packet headers, thereby avoiding the need to parse the transport packet payload data to find the program packet boundaries. As a result, decoder complexity and cost is reduced.

Generation of the sequence of transport packets 90, 92..98 is performed by the transport level packetizer 53 of Figure 2. Although Figure 3 illustrates the packetization of a single program stream, it is understood that the transport level packetizer 53 of Figure 2 separately packetizes each of the program streams input on lines 52, 54..56 in the manner illustrated in Figure 3 to produce a respective sequence of transport packets for each program stream.

Figure 4 illustrates the general content and arrangement of a program packet 102 in accordance the present invention. As shown, the program packet 102 comprises a fixed length program packet header 104 followed by a variable length payload section 106 that comprises the coded video or audio of a respective elementary stream. The program packet header 104 begins with a program packet start code 108 which, in the present embodiment, comprises a 24 bit pattern that is the same for every program packet. Hardware operating upon a program stream can use the program packet start code 108 to identify the start of each program packet. The start code 108 is followed by a program packet prefix 110 that may contain information related to the elementary stream data carried in the payload section 106. For example the prefix 110 may contain information concerning the transmission priority or encryption status of the elementary stream data in the payload section 106. The program packet prefix 110 is followed by a program packet length field 112 which specifies the total number of bytes in the program packet. As mentioned above, program packets may be of variable length.

Figure 5 illustrates the general content and arrangement of a transport packet 114 in accordance with a preferred embodiment of the present invention. As shown, the

transport packet 114 begins with a header section that comprises a transport packet start code 116 and a transport packet prefix 118. In the present embodiment, the transport packet start code 116 is an 8 bit pattern that is the same for all transport packets. Because transport packets are fixed-length packets and are arranged consecutively in a transport stream, the packet start codes will appear repetitively in an incoming transport stream at a constant rate. Accordingly, the packet start codes can be used for synchronization purposes at a decoder.

The transport packet prefix 118 may contain a number of control indicators 121, such as packet priority, error status and encryption mode indicators, relevant to the data carried in the payload section 120 of the transport packet. The transport packet prefix 118 also contains a Packet_ID (PID) field 126, which in the present embodiment comprises a 13 bit field. As mentioned above, a unique Packet_ID (PID) is assigned to each program stream. The PID assigned to a given program stream is inserted into the PID field 126 of each transport packet that carries a portion of that program stream. At a decoder, a given program stream, and ultimately the elementary stream data carried in that program stream, can be recovered from an incoming transport stream by simply extracting every incoming transport packet that carries the PID associated with that program stream. A group of related elementary streams (e.g. audio, video etc.) can be extracted to reproduce a complete "program."

In accordance with the present invention, the transport packet prefix 118 further comprises a transport adaptation field flag 122 and a program packet header indicator flag 124, each of which comprise a single bit in the present embodiment. The transport adaptation field flag 122 is used to indicate the presence or absence of an adaptation field within the payload section 120 of the transport packet 114. An adaptation field may be "opened" in any transport packet to provide a convenience "window" for carrying information of relevance to either the transport

stream or the program stream data carried in the payload section of that transport packet. For example, an adaptation field can be used to carry timing information, encryption related information, etc. In accordance with the packet alignment method of the present invention, an adaptation field can also be used for byte-stuffing purposes in order to "fill out" the payload section of a transport packet.

The program packet header indicator flag 124 is used to indicate whether the payload section 120 of the transport packet 114 begins with the header of a program packet. As explained above, in accordance with the present invention, the first byte of the header of any program packet always occupies the first byte of the payload section of a transport packet. A decoder can parse the header of each incoming transport packet and use the program packet header flag 124 to identify program packet boundaries as it extracts program stream data from the payload sections of the incoming transport packets.

Figure 6 illustrates the general content and arrangement of a transport packet 130 that contains an adaptation field. As shown, an adaptation field comprises an adaptation field header 132 and an adaptation field data region 134. The adaptation field header 132 comprises a one-byte adaptation field type specifier 136 and a one-byte adaptation field length specifier 138. The type specifier 136 specifies what type of data follows in the data region 134 of the adaptation field. As mentioned above, an adaptation field can be "opened" in a transport packet to carry information related to the transport stream or the particular program stream data carried in the payload section of that transport packet, such as timing or encryption related information. Thus, a number of different adaptation field types may be defined. In accordance with the present invention, the data region 134 of an adaptation field may also be used for byte-stuffing purposes in order to "fill-out" the payload section of a transport packet with stuffing-bytes whenever the last segment of a program packet does not

fill the entire payload section. Stuffing-bytes can be appended to the data region 134 of an adaptation field that already contains other transport or program stream related information, or if stuffing-bytes are needed in a transport packet that otherwise would not contain an adaptation field, a "stuffing bytes only" adaptation field type may be defined in which the data region 134 carries only stuffing bytes. In the present embodiment, an adaptation field type value of 00 (hex) is used to indicate the presence of a "stuffing bytes only" adaptation field. A stuffing-byte is a byte having a pre-determined pattern. For example, the bit pattern '1111 1111' can be used for each stuffing byte.

The adaptation field length specifier 138 is used to specify the overall length of the adaptation field. When the adaptation field is used for byte-stuffing in accordance with the method of the present invention, the length of the adaptation field reflects the number of stuffing bytes added to the data region 134 in order to "fill out" the payload section of the transport packet.

Figure 7 is a flow diagram illustrating further details of the operation of the encoder of Figure 2 and a preferred embodiment of the method of the present invention. In particular, the flow diagram illustrates the operation of the transport level packetizer 53 in generating a sequence of transport packets from a given program stream. It is understood, however, that the transport level packetizer 53 will perform the following steps separately for each program stream received on lines 52, 54..56 in order to produce a respective sequence of transport packets for each program stream.

At step 140, a program packet of the program stream under consideration is received by the transport level packetizer 53. At step 142, the header of a first transport packet is generated. Because the first packet will carry the first byte of the program packet header in its payload, the program packet header flag (PPH) in the transport packet header is set to a value indicative of the presence of a

program packet header. In the present embodiment, a PPH flag value of '1' indicates that a program packet header is carried in the payload section of the transport packet.

At step 144, the packetizer 53 begins constructing the payload section of the transport packet from the data of the current program packet. The first byte of the program packet will therefore be aligned with (i.e., will occupy) the first byte position of the transport packet payload. In the present embodiment, the transport packet payload is constructed from the program packet data one byte at a time. At step 146, the packetizer 53 determines whether the end of the current program packet has been reached. As described below in greater detail, in the present embodiment, the packetizer 53 identifies the end of a current program packet by detecting the beginning (i.e., the program packet start code pattern) of the next subsequent program packet in the program stream.

If at step 146 the end of the current program packet has not been detected, then control passes to step 148 where the packetizer determines whether the end of the current transport packet payload has been reached. If not, the packetizer 53 continues to fill out the payload section of the transport packet with the data of the current program packet, as illustrated by the loop formed by steps 150, 146 and 148.

Once the current transport packet payload is complete, assuming that the end of the current program packet has not been reached, control passes from step 148 to step 152 where the header of a next successive transport packet is generated. Since the next transport packet will contain all or a part of the remainder of the current program packet, and will not contain a program packet header, the PPH flag in the header of this next transport packet will be set to a value of '0'. Control then passes back to the loop formed by steps 146, 148 and 150. The packetization process thus continues using successive transport packets until the end of the current program packet is reached at step 146.

When the end of the current program packet is reached, control passes to step 156 where the packetizer 53 determines if the end of the program packet coincides with the end of the current transport packet. If so, processing of the current program packet is complete, and control passes back to step 140 where the next program packet of the program stream is received for processing in a similar manner. If, however, it is determined at step 156 that the last segment or portion of the current program packet will not occupy the entire payload section of the current transport packet, control passes to step 160.

At step 160, the packetizer 53 calculates the number of stuffing bytes that will be required in a suitable adaptation field in order to "fill-out" the entire payload section of the current transport packet. As explained above, an adaptation field may already have been created to carry other transport or program stream related information. In this case, the packetizer 53 determines the number of stuffing bytes that, along with the already existing data of the adaptation field, are needed to "fill out" the entire payload section of the current transport packet. Those stuffing bytes are then appended to the other adaptation field data at step 162 and the adaptation field length (AFL) specifier in the adaptation field header is adjusted accordingly. If the current transport packet would not otherwise have carried an adaptation field, then a "stuffing bytes only" adaptation field is generated at step 162. As explained above, in the present embodiment, a "stuffing bytes only" adaptation field has a type value of '00' (hex).

At step 164, the packetizer 53 inserts the header and data region of the adaptation field in the payload section of the current transport packet. As illustrated in Figure 3, in the present embodiment, the adaptation field precedes the remainder of the program packet in the payload section of the transport packet. In other embodiments, however, the remainder of the program packet could follow immediately after the transport packet header and the

adaptation field could then follow the program packet data to form the end of the transport packet. Finally, at step 166, the transport adaptation field flag in the header of the current transport packet is set to a value indicative of the presence of the adaptation field. Processing of the current program packet is now complete, and control passes back to step 140 for receiving the next program packet.

It should be noted that transport packets that carry the header of a program packet may also contain adaptation fields. In such a case, the first byte of the program packet header follows immediately after the adaptation field and therefore occupies the first byte of the remaining portion of the transport packet payload.

Figure 8 illustrates further details of the transport level packetizer 53 of Figure 2 and, in particular, illustrates an apparatus 200 (i.e., circuit) for carrying out the method of the present invention, as illustrated in Figure 7. The circuit 200 illustrated in Figure 8 packetizes a single program stream into a respective sequence of transport packets. As can be appreciated, the transport level packetizer 53 can either packetize each of the program streams received on lines 52, 54..56 one at a time or on a time-share basis using a single circuit 200, or alternatively, multiple implementations of the circuit 200 may be provided in the transport level packetizer 53 for packetizing the program streams in parallel.

As shown in Figure 8, a program stream, comprising a continuous succession of program packets, is received at an input 202 of the circuit 200 and provided directly to both a program packet header detection circuit 204 and a first-in-first-out (FIFO) data buffer 206. The output of the data buffer 206 is coupled to the input of a transport packet multiplexer 228 via line 220.

The program packet header detection circuit 204 detects the program packet start code pattern (PPSC) at the beginning of the header of each successive program packet and provides a "header detection" signal to a transport packet

generation controller 210 via line 208. As explained above in connection with the flow-diagram of Figure 7, the end of a current program packet is identified (step 146) by the detection of the PPSC of the next subsequent program packet.

The transport packet generation controller 210 controls the overall construction of each successive transport packet in the sequence to be generated for the program stream received on line 202. When the beginning of a current program packet is detected by the detection circuit 204, the transport packet generation controller 210 initiates the construction of a transport packet to carry a first portion of the current program packet. According to the method of the present invention, the first byte of the current program packet will be aligned with (i.e., will occupy) the first byte of the payload portion of this first transport packet.

Under the control of the transport packet generation controller 210, a transport packet header generator 212 generates a transport packet header for the first transport packet. The Packet_ID associated with the program stream received on line 202 is inserted in the PID field of the transport packet header. Additionally, since this first transport packet will carry the header of the current program packet, the program packet header indicator flag (PPH) in the transport packet header is set to a value of '1'. The data forming the transport packet header is then provided, in sequence, to the input of a transport packet multiplexer via line 216. The transport packet generation controller 210 then selects line 216 for output on line 232 using the data selection multiplex control lines 230. Thus, the header of the transport packet is output on line 232.

Assuming no adaptation field is to be carried in the current transport packet, when the end of the transport packet header begins to pass through the multiplexer 228, the controller 210 loads a transport packet payload size counter 222 with a value equal to the total fixed length, in bytes, of the transport packet payload section. As explained above,

transport packets have a fixed length. As the last bit of the transport packet header reaches the output 232 of the transport packet multiplexer 228, the transport packet generation controller enables the output of the data buffer 206 and switches the output 232 of the multiplexer 228 to line 220 using selection control lines 230. The data of the current program packet, beginning with the first byte of the program packet header, therefore immediately follows the transport packet header on output line 232. As successive bytes of the current program packet are output from the buffer 206, the transport packet generation controller 210 decrements the payload length byte count in counter 222. When the counter 222 reaches zero, assuming that the end of the current program packet has not been detected, the transport packet controller 210 disables the output of the buffer 206 and switches the output of the multiplexer 228 back to line 216. The controller 210 then initiates the generation of a next subsequent transport packet. As before, the transport packet header generator 212 generates the transport packet header and outputs the header, bit-by-bit, on line 216. Because this next subsequent transport packet will contain a remaining portion of the current program packet, and will not contain a program packet header, the program packet header indicator flag (PPH) will be set to a value of '0'.

If the current transport packet is to contain an adaptation field in order to carry transport or program stream related information, an adaptation field generator 214 begins generating the header and data region of the adaptation field and provides the adaptation field data to a third input of the transport packet multiplexer 228 via line 218. Depending on the adaptation field type, as specified in the type field of the adaptation field header, a respective one of a plurality of counters 226 is loaded with a value indicative of the length of the adaptation field being generated. When the end of the transport packet header reaches the output 232 of the multiplexer 228, the transport

packet controller 210 will switch the output 232 of the transport packet multiplexer 228 to line 218 to begin outputting the adaptation field data on line 232 immediately after the transport packet header. As each byte of the adaptation field is output on line 218, the respective adaptation field length counter 226 is decremented. When the appropriate counter 226 reaches zero, the transport packet generation controller 210 loads the transport packet payload size counter 222 with a value equal to the total length of the transport packet payload minus the length of the adaptation field; that is, the counter 222 is loaded with a value equal to the remaining length of the payload portion that will follow the adaptation field. The controller 210 then enables the output of the data buffer 206 and switches the output of the transport packet multiplexer 228 to line 220. Successive bytes of the remaining portion of the current program packet are output on line 220 until the counter 222 again reaches zero. As noted above, the first transport packet that carries the program packet header may also contain an adaptation field, in which case, the first byte of the program packet header follows immediately after the adaptation field and therefore occupies the first byte of the remaining portion of the transport packet payload.

The circuit 200 continues to function as described above, generating successive transport packets that carry successive portions of the current program packet, until the PPSC of a next subsequent program packet in the program stream is detected, i.e., until the end of the current program packet is detected. The buffer 206 is of sufficient size to provide the controller 210 with sufficient time to calculate the number of stuffing bytes, if any, that will be needed to fill-out the payload section of the last transport packet to be generated for the current program packet. At the appropriate time, the adaptation field generator loads a value equal to the required number of stuffing bytes into a respective counter 224 and begins generating the adaptation field as described above. Once the end of the adaptation

field reaches the multiplexer 228, the controller switches the output of the multiplexer 228 to line 220 to output the last segment or portion of the current program packet. When this is complete, the circuit 200 repeats the process for the next program packet in the program stream. Thus, the first byte of each program packet will always be aligned with, i.e., will always occupy, the first byte of the payload section of a respective transport packet.

Figure 9 is a block diagram of a decoder apparatus 170 for parsing an incoming transport stream to extract one or more elementary streams from the transport stream. For example, in a subscription television system, the decoder 170 may be employed to select a particular program, i.e., related group of elementary streams, for output at a subscriber location. Thus, the decoder 170 of Figure 9 may be used to implement the decoder 40 of Figure 1.

As shown in Figure 9, an incoming transport stream is provided via line 172 to a transport stream demultiplexer 176. A user's program selection is provided to the demultiplexer 176 via line 174. As explained above, each transport packet carries the Packet_ID (PID) associated with the program stream data carried in that transport packet. A different PID is associated with each of the different program streams carried in the transport stream. A mapping table may be provided in the decoder that identifies, for each user selectable program, the PIDs associated with the program streams that carry the elementary streams that "make-up" that program. Thus, when a user selects a given program for output, the demultiplexer 176 can access this mapping information to identify the PIDs of the transport packets that carry the program stream data for each of the elementary streams (e.g., video, audio, etc.) that "make-up" the selected program. Such a program mapping technique is incorporated in the MPEG-2 System standard (ISO 13818) as described in the aforementioned MPEG-2 Systems Committee Draft.

Once the PIDs for the selected program are identified, the demultiplexer 176 begins extracting every incoming transport packet having a PID that matches one of those listed for the selected program. For example, a subscriber may select a program that consists of elementary streams "Video 1" and "Audio 1." Assuming that the transport packets carrying the program stream data for "Video 1" each have a PID of '10', and that the transport packets carrying the program stream data for "Audio 1" each have a PID of '12', the demultiplexer 176 will extract every incoming Transport Packet having a PID of '10' or '12'.

As the transport packets that contain the program stream data for each elementary stream of the selected program are retrieved from the incoming transport stream, they are provided, in sequence, to a transport level parsing unit 182. For example, the transport packets carrying program stream data for a video elementary stream are provided to the transport level parsing unit 182 via line 178. Transport packets carrying program stream data for a related audio elementary stream are provided to the parsing unit 182 via line 180. The transport level parser 182 operates to retrieve the respective program streams from the respective transport packet sequences. Essentially, the transport level parser 182 extracts the data carried in the payload sections of each transport packet and then reconstructs the program packets of each program stream. As a result of the packet alignment method of the present invention, the transport level parsing unit 182 does not have to parse the payload sections of each transport packet to identify program packet boundaries. Rather, the parsing unit 182 can simply employ the program packet header indicators (PPH) in the headers of each successive transport packet to identify the program packet boundaries. This greatly simplifies the implementation of the transport level parsing unit 182 and reduces the overall cost of the decoder.

As the program packets of the respective program streams are reconstructed, they are provide via respective

lines 184 and 186 to a program level parser 188. The program level parser 188 operates to recover the raw elementary stream data from the respective program packets of each program stream. The audio and video elementary stream data recovered from the respective program streams are provided to respective buffers 190, 192 and then to respective decoders 194, 196 which decode the elementary stream data to produce analog video and audio signals for output to a display device (not shown).

As the foregoing illustrates, the present invention is directed to a method and apparatus for packetizing a program stream into a sequence of transport packets such that the beginning of a program packet is always aligned with the beginning of a transport packet payload. With the present invention, a decoder does not need to parse the transport packet payload data to determine program packet boundaries. Consequently, decoder cost and complexity is reduced.

As described and claimed herein, the packet alignment method and apparatus of the present invention are not limited to the particular two-level multiplexing scheme described above, including the particular packet structures and encoder and decoder implementations. Rather, the present invention may be employed in any two-level multiplexing scheme in which a first level packet stream is packetized or encapsulated within smaller, fixed-length packets of a second level packet stream. Thus, the terms "program packet" and "transport packet" are not intended to be limited to the particular packet structures described above, but are intended to encompass the respective packets of the first and second levels of any two-level packet based multiplexing scheme. For example, as explained above, the method of the present invention, as defined by the appended claims, has been substantially incorporated as part of the two-level multiplexing scheme of the MPEG-2 Systems standard in which the packets of the first multiplexing level are referred to as Program Elementary Stream (PES) packets. Accordingly, the term "program packets", as used in the appended claims, is

intended to encompass the Program Elementary Stream packets of the MPEG-2 Systems standard. It is understood therefore that changes may be made to the embodiments described above without departing from the broad inventive concepts thereof. This invention is not limited to the particular embodiments disclosed, but is intended to cover all modifications that are within the scope and spirit of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method of packetizing a data stream for transmission comprising the steps of:

(a) segmenting the data stream and inserting successive segments of the data stream into a plurality of successive program packets, each program packet having a header followed by a payload section;

(b) for each of said successive program packets:

(i) inserting successive segments of the program packet into the payload sections of a respective sequence of transport packets, the header of the program packet being aligned with the beginning of the payload section of a first transport packet in said respective sequence; and

(ii) repeating step (b)(i) until a last segment of the program packet is reached, and if the last segment of the program packet does not fill the entire payload section of a last transport packet in said respective sequence, then inserting stuffing bytes in the last transport packet until the entire payload section is completely filled.

2. The method of claim 1 wherein step (b)(ii) comprises inserting said stuffing bytes in an adaptation field of said last transport packet.

3. The method of claim 1 wherein every transport packet has a header and wherein step (b)(i) further comprises setting a bit in the header of said first transport packet to indicate that said first transport packet carries the header of said program packet.

4. The method of claim 1 wherein said successive program packets have variable length payload sections.

5. A method of packetizing a data stream for transmission comprising the steps of:

(a) segmenting the data stream and inserting successive segments of the data stream into a plurality of successive program packets, each program packet having a header followed by a payload section;

(b) receiving a first one of the plurality of successive program packets, the first program packet defining a current program packet;

(c) inserting successive segments of the current program packet into the payload sections of a respective sequence of transport packets, the header of the current program packet being aligned with the beginning of the payload section of a first transport packet of the respective sequence;

(d) repeating step (c) until a last segment of the current program packet is reached, and if the last segment of the current program packet will not fill the entire payload section of a last transport packet in said respective sequence, then inserting stuffing bytes in the last transport packet along with the last segment of the current program packet until the entire payload section is filled; and

(e) receiving a next successive one of the plurality of program packets, re-defining the current program packet as said next successive program packet, and repeating steps (c) and (d).

6. The method of claim 5 wherein step (d) comprises inserting said stuffing bytes in an adaptation field of said last transport packet.

7. The method of claim 5 wherein every transport packet has a header and wherein step (c) further comprises setting a bit in the header of said first transport packet to indicate that said first transport packet carries the header of said current program packet.

8. The method of claim 5 wherein said successive program packets have variable length payload sections.

9. An apparatus for packetizing a data stream for transmission comprising:

means for segmenting the data stream and inserting successive segments of the data stream into a plurality of successive program packets, each program packet having a header followed by a payload section;

means for receiving one of the plurality of successive program packets;

means for inserting successive segments of the received program packet into the payload sections of a sequence of transport packets such that the header of the current program packet is aligned with the beginning of the payload section of a first transport packet of the sequence; and

means for inserting stuffing bytes in a last transport packet of the sequence if a last segment of the program packet will not fill the entire payload section of said last transport packet.

10. The apparatus of claim 9 further comprising means for generating an adaptation field for insertion in the payload section of a transport packet.

11. The apparatus of claim 10 wherein said means for inserting stuffing bytes is adapted to insert said stuffing bytes in an adaptation field inserted in said last transport packet.

12. The apparatus of claim 9 wherein every transport packet has a header and wherein said apparatus further comprises means for setting a bit in the header of said first transport packet to indicate that said first transport packet carries the header of said received program packet.

13. An apparatus for packetizing a program stream into a continuous sequence of transport packets, wherein said

program stream comprises a continuous sequence of program packets and each of said transport packets has a fixed length, said apparatus comprising:

means for receiving one of the successive program packets;

means for inserting successive segments of the received program packet into the payload sections of a sequence of transport packets such that the header of the current program packet is aligned with the beginning of the payload section of a first transport packet of the sequence; and

means for inserting stuffing bytes in a last transport packet of the sequence if a last segment of the program packet will not fill the entire payload section of said last transport packet.

14. The apparatus of claim 13 further comprising means for generating an adaptation field for insertion in the payload section of a transport packet.

15. The apparatus of claim 14 wherein said means for inserting stuffing bytes is adapted to insert said stuffing bytes in an adaptation field inserted in said last transport packet.

16. The apparatus of claim 13 wherein every transport packet has a header and wherein said apparatus further comprises means for setting a bit in the header of said first transport packet to indicate that said first transport packet carries the header of said received program packet.

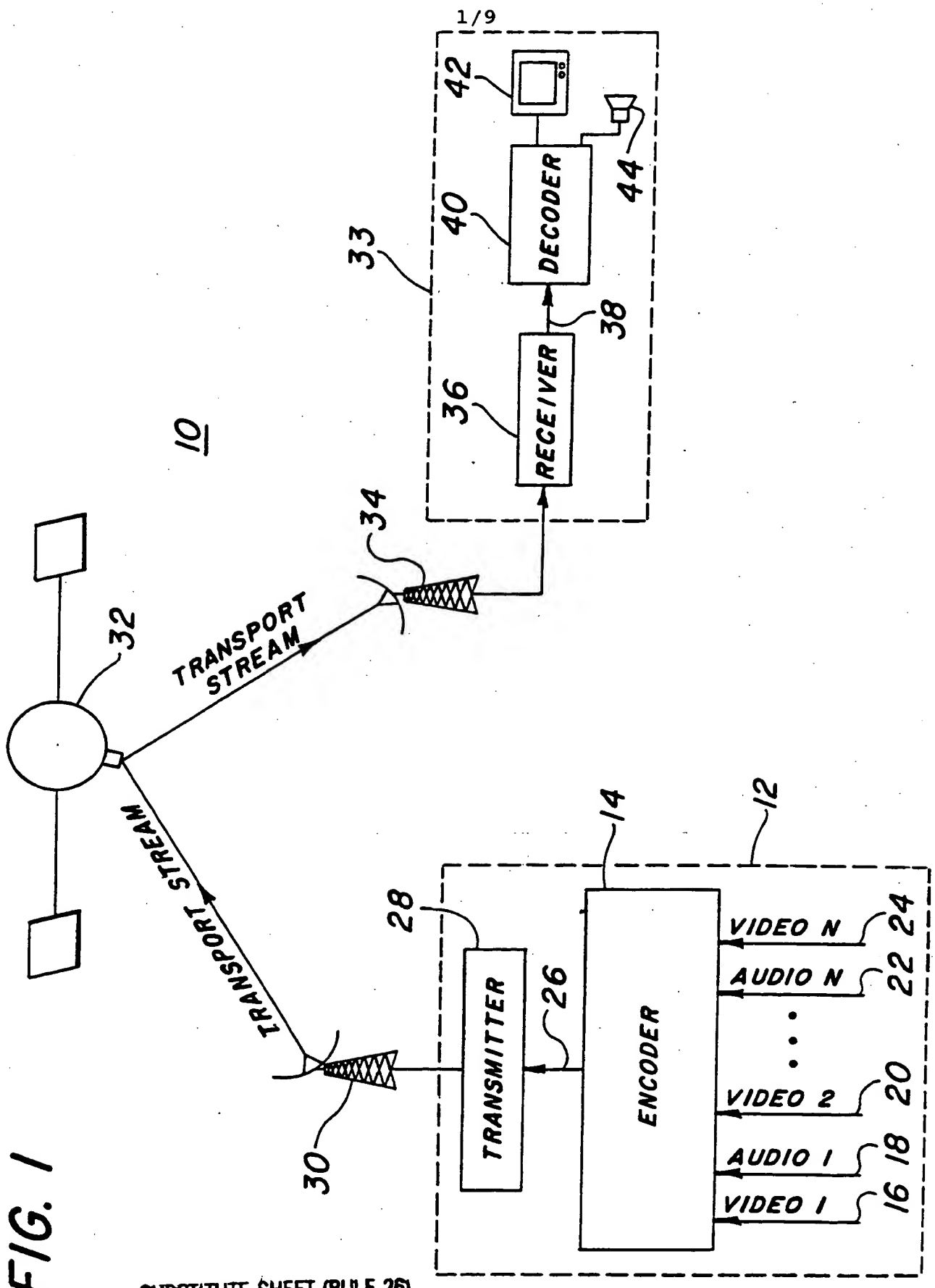


FIG. 1

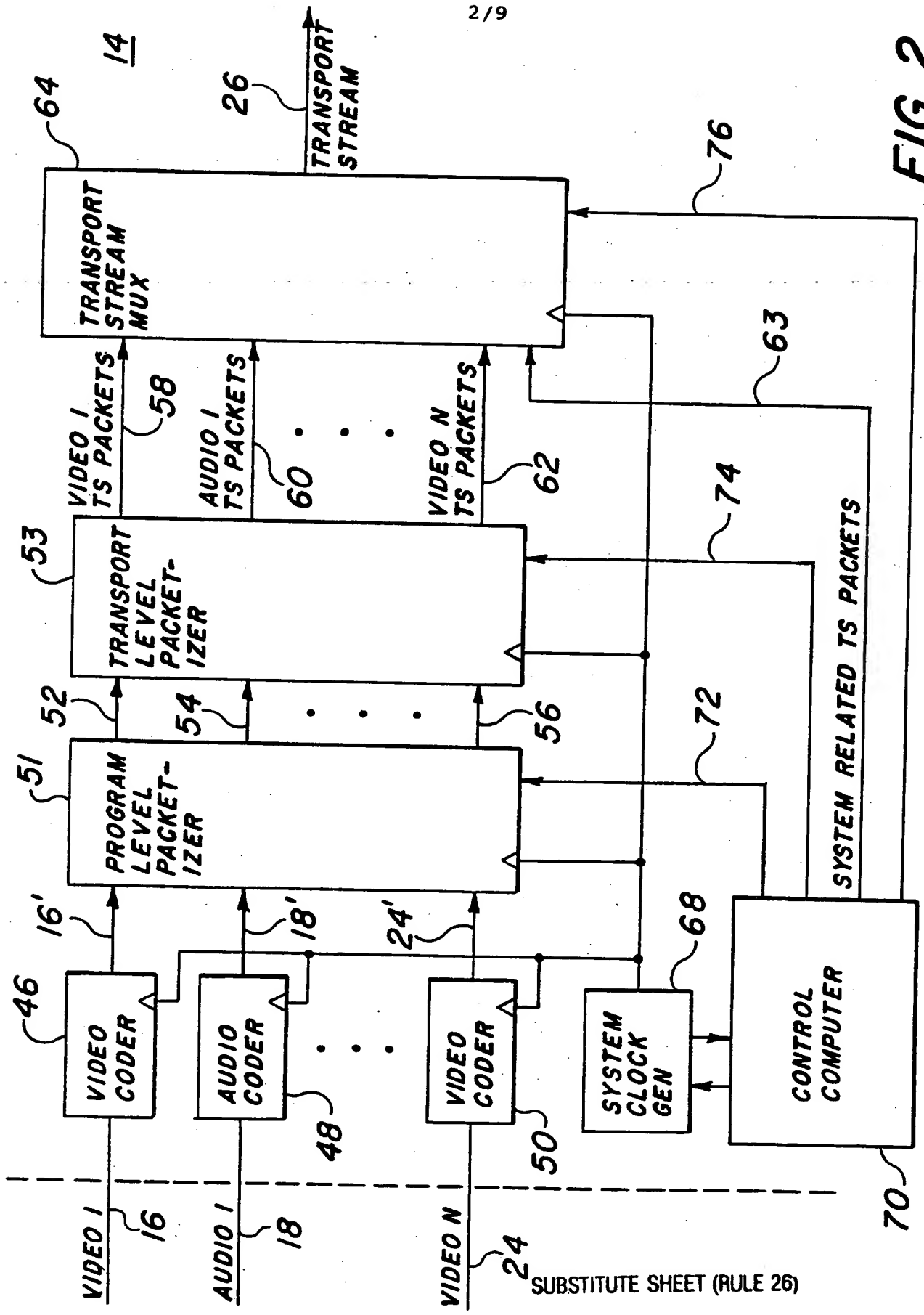

14
64
26
TRANSPORT STREAM
TRANSPORT STREAM MUX
76
63
VIDEO 1 TS PACKETS
58
AUDIO 1 TS PACKETS
60
VIDEO N TS PACKETS
62
53
TRANSPORT LEVEL PACKET-IZER
74
SYSTEM RELATED TS PACKETS
52
54
56
51
PROGRAM LEVEL PACKET-IZER
72
16'
18'
24'
68
46
VIDEO CODER
AUDIO CODER
48
VIDEO CODER
50
SYSTEM CLOCK GEN
CONTROL COMPUTER
70
VIDEO 1
16
AUDIO 1
18
VIDEO N
24

FIG. 2

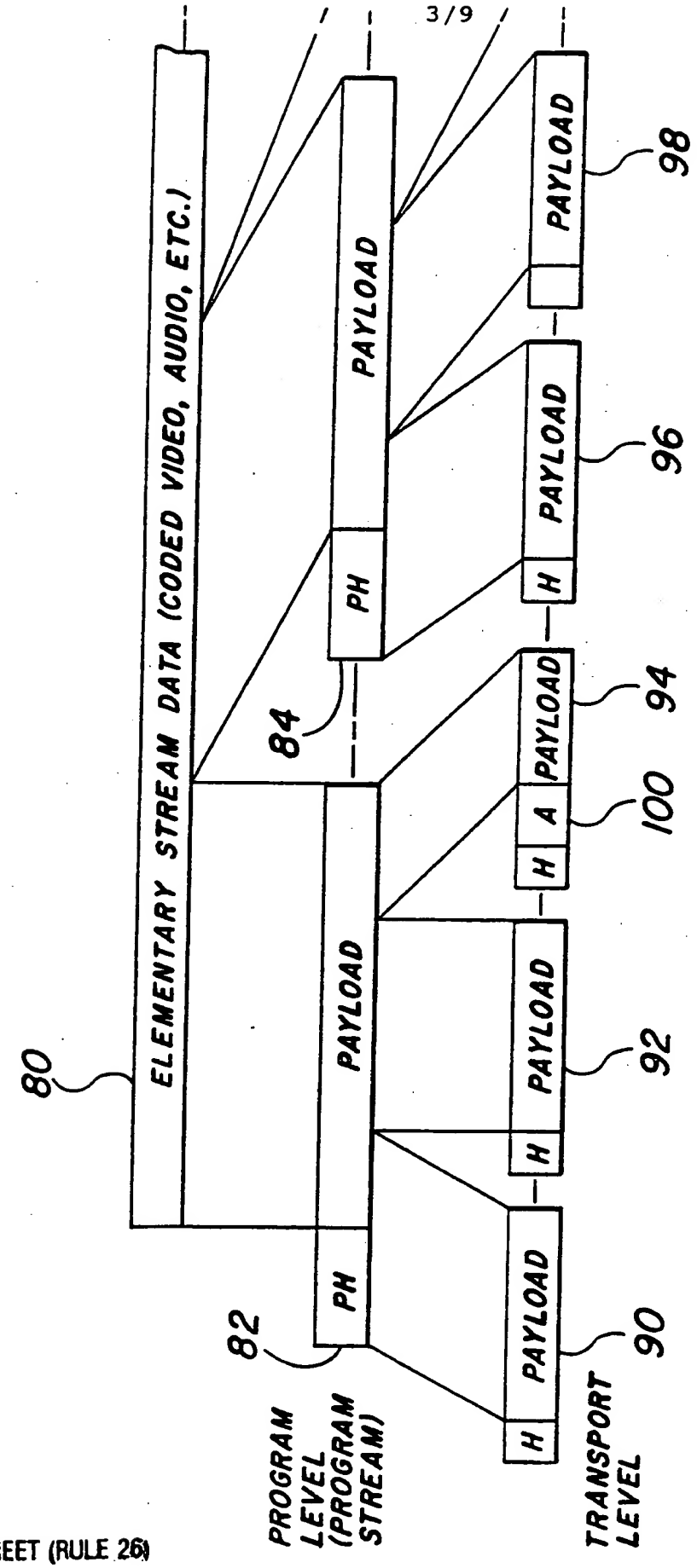


FIG. 3

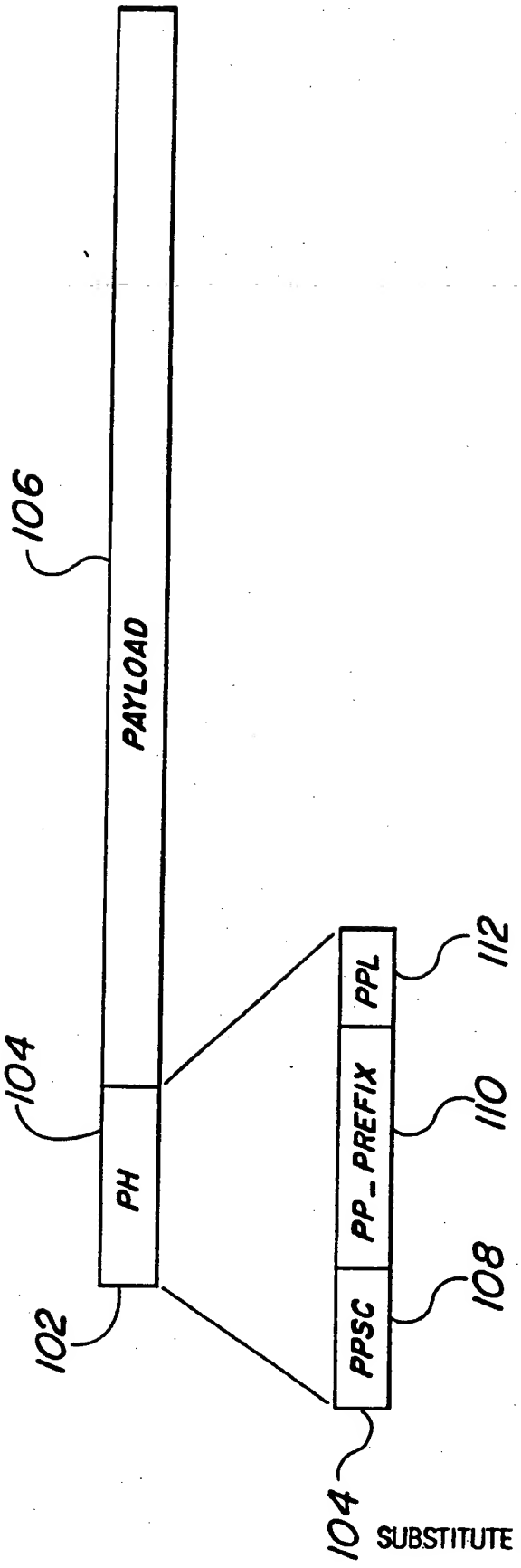


FIG. 4

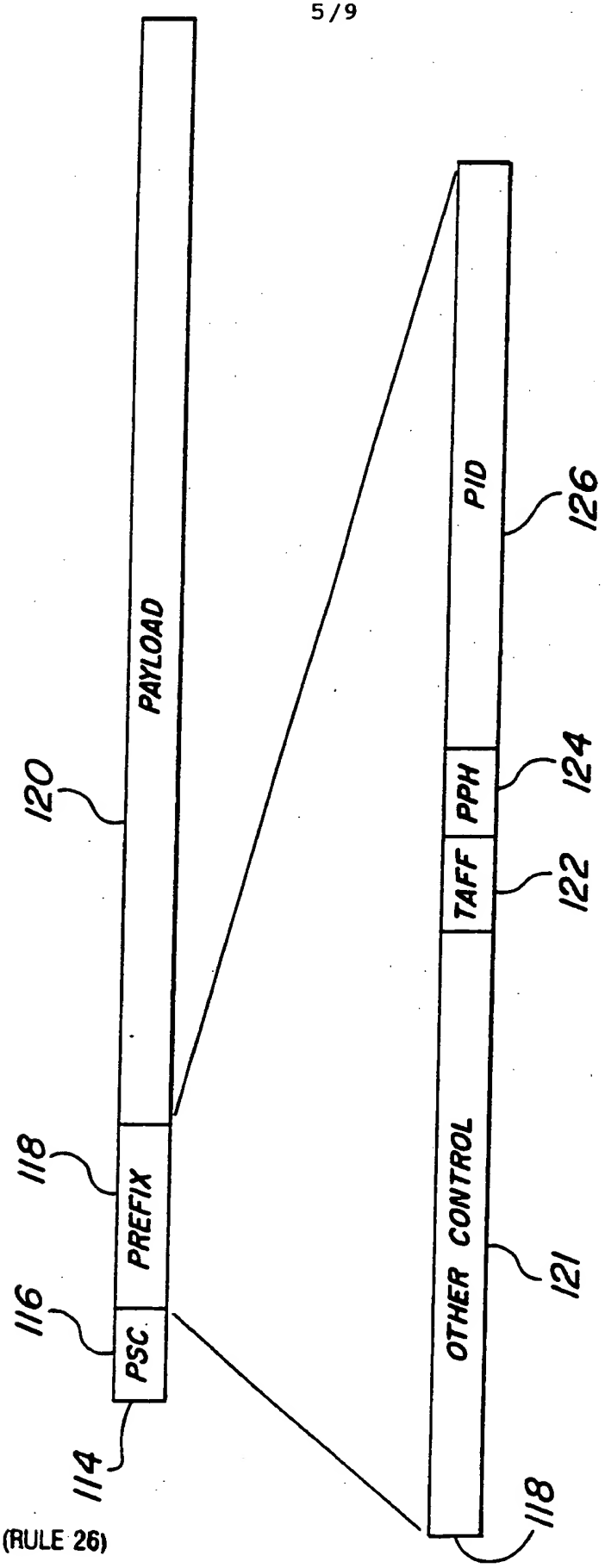


FIG. 5

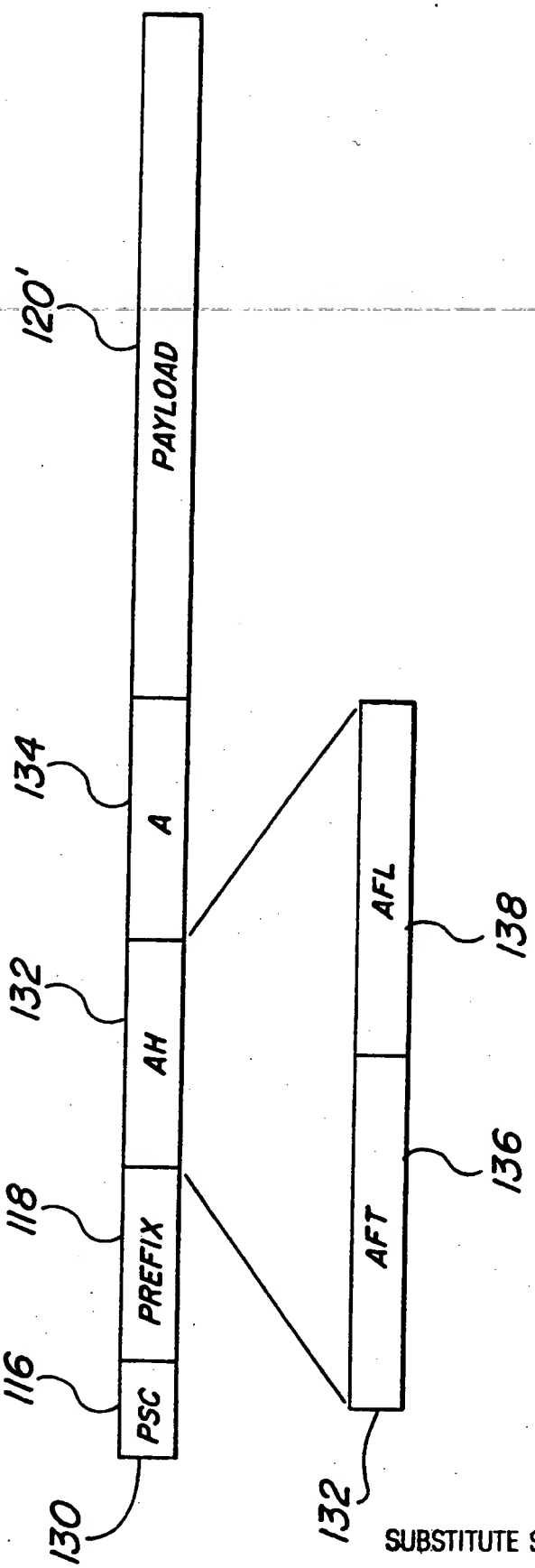
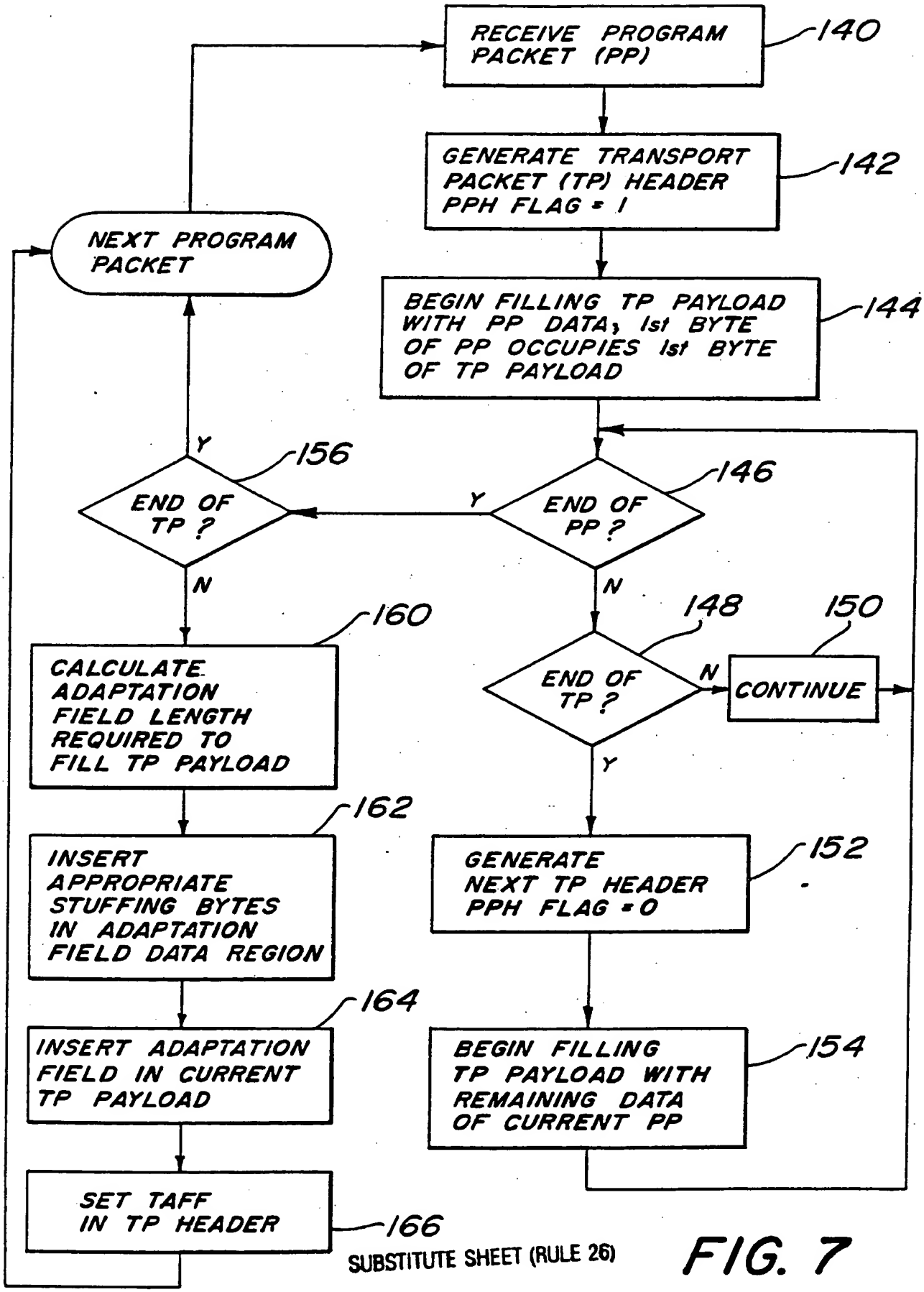

120'
PAYLOAD
134
A
132
AH
118
PREFIX
116
PSC
130
AFL
138
AFT
136
132

FIG. 6



SUBSTITUTE SHEET (RULE 26)

FIG. 7

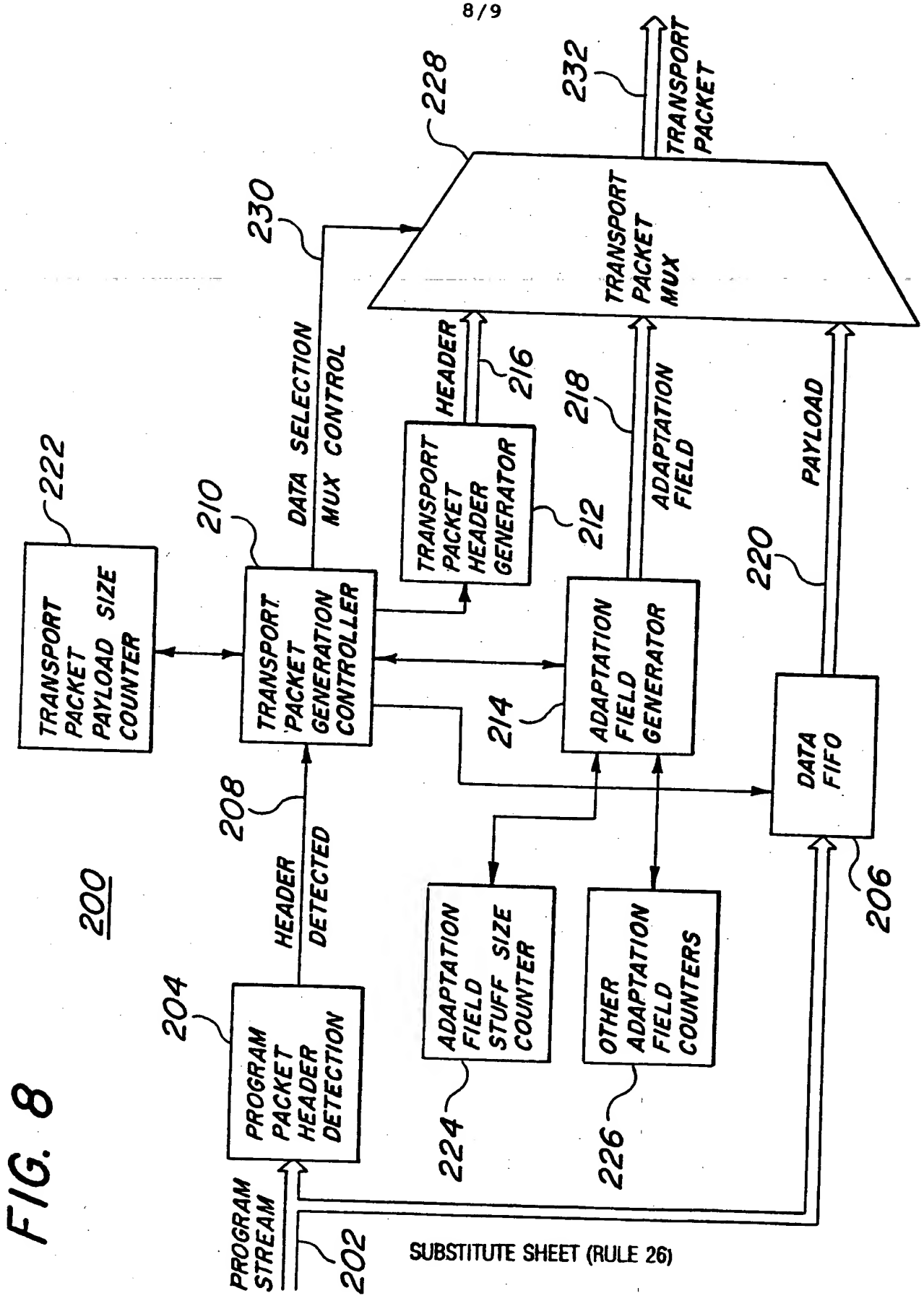

200
PROGRAM STREAM
202
PROGRAM PACKET HEADER DETECTION
204
HEADER DETECTED
208
TRANSPORT PACKET GENERATION CONTROLLER
210
TRANSPORT PACKET PAYLOAD SIZE COUNTER
222
DATA SELECTION MUX CONTROL
230
TRANSPORT PACKET HEADER GENERATOR
212
HEADER
216
ADAPTATION FIELD GENERATOR
214
ADAPTATION FIELD
218
PAYLOAD
220
DATA FIFO
206
ADAPTATION FIELD STUFF SIZE COUNTER
224
OTHER ADAPTATION FIELD COUNTERS
226
TRANSPORT PACKET MUX
228
TRANSPORT PACKET
232

FIG. 8

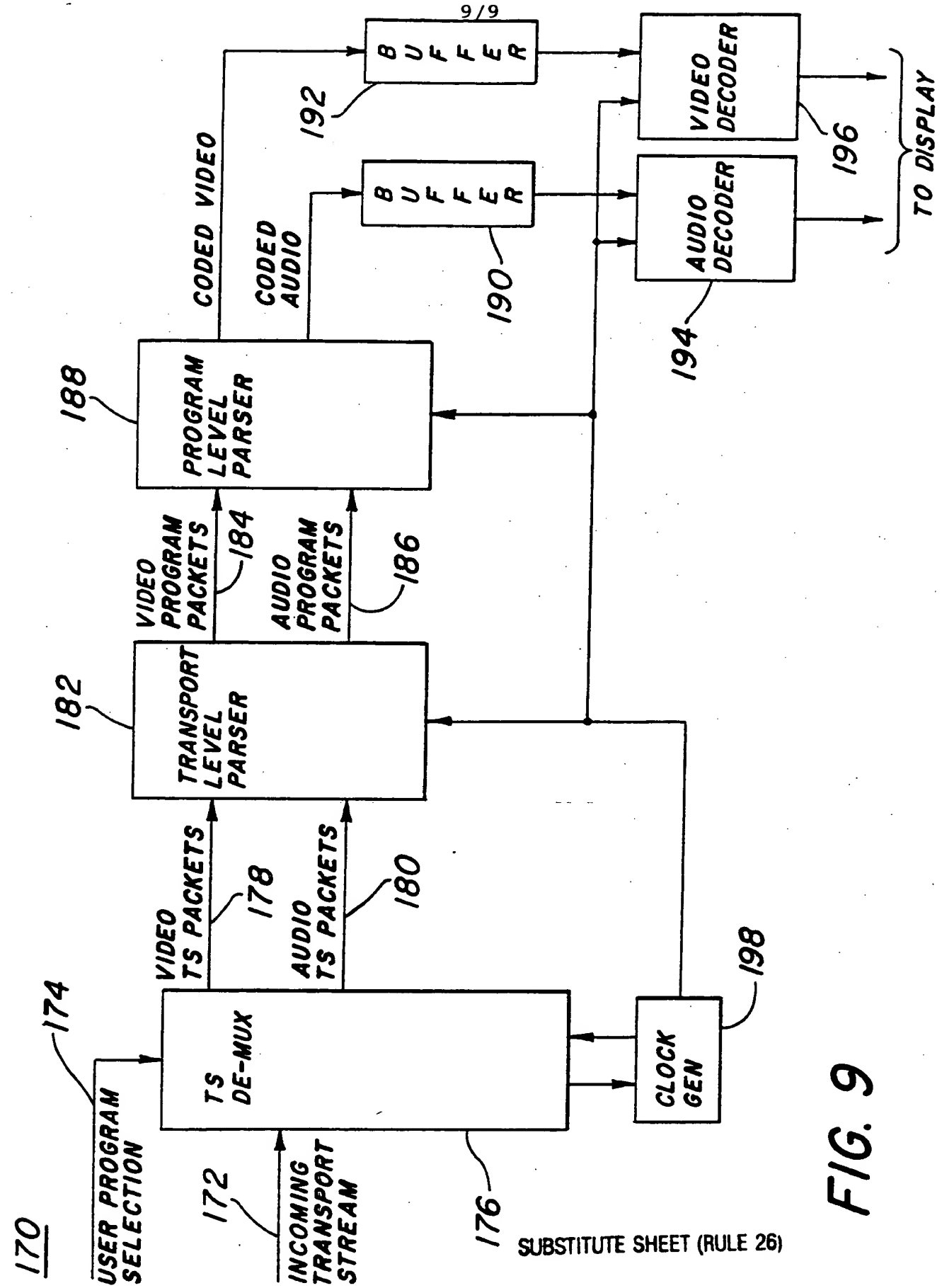


FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US94/07864

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H 04J 3/02

US CL :370/60, 60.1, 79, 82, 94.1, 99

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/60, 60.1, 79, 82, 94.1, 99

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 5,237,569 (SEKIHATA ET AL.) 17 August 1993, col. 7, line 60, to col. 8, line 22, and Fig. 11.	1-16

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search

03 OCTOBER 1994

Date of mailing of the international search report

OCT 20 1994

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Chau T. Nguyen

Telephone No. (703) 308-5340

Form PCT ISA/210 (second sheet)(July 1992)*